VIP
Library

VIP Disk-ZAP™

Diskette Repair Utility

Operator's Manual

Softlaw

For the TRS-80 Color and TDP System 100 Personal Computer

Softlaw Corporation 9072 Lyndale Ave. So.
Minneapolis, Minnesota 55420 612/881-2777

T A B L E O F C O N T E N T S

Disk repair is something that each of us needs. Sooner or later we drop a disk, or our operating system fails, leading to disk failure. Up on the screen pops a dreaded message such as "I/O ERROR". The file is lost. It could be a grocery list, or the only copy of a valuable customer order database. Now, with the VIP Disk-ZAP, anybody, from non-programmer to experienced assembly language programmer, can fix repairable disk errors in minutes and return to work with the fixed disk.

The VIP Disk-ZAP is a fast machine-code program for use with the standard Color Computer disk operating system. It was designed with the non-programmer in mind. The user is guided through every step with easy-to-use menus and full message prompting. An attractive and well organized screen display makes the program even simpler to use. It features a split screen, dual cursor format which allows total control and monitoring of every step. Commands are all single key nmemonics which are easy to use and remember. Every modification to be made to the disk or file is verified with you before being finally made so you don't have to worry about making careless mistakes.

The many features of the VIP Disk-ZAP make it the <u>only</u> disk repair utility you'll ever need for your Color Computer. You can:

* Fix - simply - all repairable disk errors.
* Verify to find all errors throughout the entire disk and obtain a description of any error found.
* Move easily through the file or disk by sector or track.
* Repair individual files or the entire disk at once.
* Address any disk drive; work on any granule, track or sector.
* Modify textfile or disk using ASCII or hexidecimal input.
* Copy sectors to any drive and track.
* Locate specific numeric or ASCII strings for modification.
* Print sectors to any RS-232 device at any of six different baud rates.

PART I: How the VIP Disk-ZAP Works

About This Manual

Disk repair requires two things: a good repair utility and knowledge. The first we have provided with the VIP Disk-ZAP program. The second you will obtain from experience, from reading reference works, and from this manual. There is no general work on the market which deals with data recovery; nor is there any book or article dealing with data recovery with the Color Computer. This manual cannot pretend to teach you any but the most simple concepts. Until there is some work devoted to the subject you will have to depend on your disk manual, technical reference manuals, and on the rudiments taught in this manual. Still, most disk errors are easily fixed, so don't despair.

This Manual is divided into two sections. The first part tells you how the VIP Disk-ZAP works, detailing its menus, commands and screen displays. After the loading instructions are explained, the commands of the Master Menu are discussed in a logical order. Some of these commands present a separate menu with sub-commands which are also fully discussed. Since an understanding of the basics of disk structure is required before learning how to use this program, it is highly recommended that you read the section on disk structure in Part II before learning how to use this program.

The second part of this Manual gives a general beginner's explanation of how to use the VIP Disk-ZAP to fix your disks. It includes a discussion of the structure of the disk, the basic concepts of disk repair, a summary of common errors which can affect disk access and their causes, fixes for those errors, and instructions on how to rebuild files and the directory, manually backup a disk, and allocate a granule containing an unrepairable sector.

This manual assumes that you are familiar with the hexidecimal number system. For your convenience a cross-reference chart between decimal and hexidecimal numbers is provided in the Appendix. On thing you should be aware of is that in many computer usages, items are counted beginning with the number zero. Thus, there are 35 tracks on each disk, numbering from 0 to 34. When referring to items numbered beginning with zero, the convention is to call the number assigned "relative". Thus, the 35th track is relative track 34; the first track is relative track 0. This can be awful confusing at times, so be careful.

Loading the VIP Disk-ZAP from the Diskette

U N P L U G   Y O U R   J O Y S T I C K S !

Mount the VIP Disk-ZAP master diskette in Drive 0, type LOADM"ZAP" and press <ENTER>. The program will automatically load and execute. A billboard will be displayed while the program is loading and prior to execution. When the program has executed, the Master Menu will appear.

System Requirements

The VIP Disk-ZAP program is only designed to be used with the standard Color Computer Tandy operating system. It will not work on other operating systems for the Color Computer such as FLEX or OS-9, or on files created using such operating systems.

Getting Started

Now that you have read the section in Part II explaining disk structure, put a disk which you don't care about ruining (or write protect a disk) and place

it in drive zero and let's get started. After the program has executed, the Master Menu will appear displaying a selection of single key entry commands. The following is a list of the commands available as they appear on the Master Menu with a brief description of their functions:

Directory <A>: Pressing this key will elicit the directory for the drive specified.

Disk Zap <D>: This command allows repairs to be made to the disk as a whole. A separate menu controls Disk Zap access.

Exit Program <BREAK>: Press this key to exit the program and return to BASIC.

File Name <N>: With this command you are told the name of the last file accessed.

File Zap <F>: This command allows access to particular files for repair. A separate menu appears to prompt the user in File Zap access.

Kill <K>: This command allows you to kill any file on the disk.

Read Sectors <R>: With this command you can load several sectors into memory for copying.

Toggle X-Late <X>: This command toggles between the true ASCII and original Color Computer display.

Toggle Base <SHIFT><@>: With this command you toggle between the hexidecimal and decimal number systems for selecting drive, track and sector numbers.

Toggle ASCII <CLEAR>: This key toggles input in the Modify and Locate functions between ASCII and hexidecimal.

Verify Disk <V>: This command causes the program to verify the entire disk, stopping sequentially at any errors found.

Write Sectors <W>: This command is used to Write the sectors read into the memory with the Read Sectors command to another location.

At the bottom of the screen on a Command line is the prompt "REQUEST?" which asks for input of one of the letter commands. All commands must be entered in the uppercase mode (discussed below).

Toggle Commands: X-Late, Base, and ASCII

Toggle Base. At the right of the screen is the flag "UXDEC". "DEC" is the "Toggle Base" flag which, when "DEC", stands for "decimal" and indicates that numeric input into the system when indicating the drive, track or sector to be accessed must be in the decimal number system (base 10). Decimal may not be used to input in the modify functions. Input is restricted to hexidecimal numbers. (See Modify, discussed in the File Zap section.) You may change your input to hexidecimal by pressing <SHIFT><@> to Toggle the Base. When hexidecimal has been toggled the "DEC" flag will change to "HEX". Pressing it again toggles back to decimal. The number base may ONLY be toggled from the Master Menu.

When in either base, you have the option of forcing input to be in the alternative base by preceding the number by a character code. If you are in the DEC base, you may input hexidecimal numbers by preceding each number with a "$" symbol. When in the HEX base you may input decimal numbers by preceding each number with the

"&" symbol. The following example shows use of these symbols with the Enter Track command from the Disk Zap menu (see below):

EXAMPLE: TRACK, SECTOR? &17, 03

The above example assumes that you are in the HEX base and want to input the drive and track numbers in decimal for the directory track (see Part II). Track number 17 is preceded by the "&" symbol to allow decimal input. There is no need to precede the sector number 03 with the ampersand since the hexidecimal and decimal numbers for 03 are the same.

When in the decimal base, the numbers displayed in the DRV, TRK and SEC indicators, the GRAN indicator when accessing sectors (discussed below) and the relative BYTE indicator in the Modify functions discussed below), will be in the decimal base. Otherwise they will be in the hexidecimal base.

Toggle X-Late. The "X" of the "UXDEC" flag is the "Toggle X-Late" ("translate") flag which is either present or not present. Its presence is toggled by pressing <X> from the Master Menu ONLY. When present, as it is when the program is first executed, it indicates that the ASCII display of files will be translated into the Standard ASCII Character Set (see Appendix and the discussion of the screen display below); when it has been toggled to "U DEC" without the "X", this indicates that the ASCII display will show the file in the character set used by the Color Computer display generator chip.

Toggle ASCII. Sometimes, you will be required or you will desire to input ASCII characters (i.e., words and characters) instead of numbers. ONLY the Modify (see Disk and File Zap commands) and Locate (see Disk Zap commands) allow ASCII input. In the Modify function you must choose specifically between making modifications in ASCII or in hexidecimal (never

decimal). The system begins with ASCII input being enabled. Hexidecimal input is enabled by using the Toggle ASCII command, <CLEAR>, which may be done at any time. When ASCII is enabled, as it is when the program is first entered, it is indicated on the screen by the presence of the flag "ASC" at the far right of the screen next to the "UXDEC" flag. Even when the ASCII option has been disabled, the system will accept all command letters.

Toggling Upper/Lower Case. When the program is first executed all keyboard entry is in upper case. To input both lower and upper case press <SHIFT><0>. This toggles case status just as in BASIC. Case status may be toggled at any time when using the program. A case mode indicator is placed in the first position of the "UXDEC" flag. The "U" indicates that the uppercase mode is in use. When the case status is toggled to lower/upper case the "U" changes to an "L". NOTE that all commands must be input in upper case.

<BREAK>: Exit

The <BREAK> key is the general exit command and has two distinct uses: exiting from a command to the Master Menu and exiting from the Master Menu out of the program altogether returning to BASIC. While using the commands and sub-commands of the VIP Disk-ZAP, you may exit to the Master Menu at any time by pressing <BREAK>.

However, pressing <BREAK> while in the Master Menu will initiate an exit from the VIP Disk-ZAP. It is very convenient to exit from a utility program to either test your handiwork or get back to work with what you have fixed. You may only exit this program from the Master Menu. When you press <BREAK> to initiate the exit the system will prompt on the command line "Do you really wish to exit?" Press <Y> to exit to BASIC; if you press any other key the exit will be aborted and the

"Request:" prompt will reappear awaiting your further command.

The remaining commands on the Master Menu will be discussed in logical divisions. The first section, entitled "File Determination," will be devoted to calling up the disk directory and determining which files are accessible. This section includes the Directory and File Name commands. The second section, entitled "File and Disk Access," consists of the commands used for actual file and disk access, error checking and modification. This section includes the Verify, Read and Write Sectors, and the File Zap and Disk Zap commands. The File Zap and Disk Zap commands each call up a separate menu for manipulation of the individual files and the disk as a whole. A final section will discuss the KILL command.

File Determination

The following sections deal with determining which files to access. It includes the Filename and Directory commands.

<F> Filename

The Filename command may be used from the Master Menu to determine the last filename used. When you press <N> the screen will clear and at the bottom will appear the drive, track and sector numbers of the last file accessed. Below that will appear the prompt "Filename:" followed by the name of the last diskfile accessed, including its extension and drive number. Press <BREAK> to exit to the Master Menu for further work on that or another file.

<A> Directory

This command directs the system to display the directory of the disk to be repaired. After you press <A> the Master Menu disappears and you are prompted wit "Drive:", a request for the drive number of the drive containing the disk whose directory is to be displayed. Upon entering the number (0, 1, 2, or 3), or <ENTER> for the last drive accessed, the system will automatically display a directory of the disk requested. The directory for the specified drive will be displayed in a two-column format with the drive number and number of free granules on the disk shown on the top line of the display. Each entry in the directory will be followed by a number indicating the number of granules allocated to the file on the disk, and the letter "A" for ASCII or "B" for binary to indicate the nature of the file.

Occasionally an asterisk or an "R" may also appear in the directory after the filename. These characters indicate that there is an error in the granule allocation of the file which will prohibit access of that file. The errors are of two types. In one instance the granule allocation table contains a number other than the allowed (in hex) 00-43, FF, or C0-C9. This error elicits an asterisk by the bad file name. The second error is caused by a granule allocation byte referring to itself as the next granule in the file, causing the system to repeatedly load that granule, believing that the file has over 68 granules. Because of this repeated access of the same granule, this error is assigned an "R" in the directory. Whenever you encounter either of these errors when Verifying the disk a BAD FILE error prompt will be displayed. The system will not allow you to load any such damaged sector for repair until the directory is repaired. (To effect repair see the discussion of the structure of the directory and the discussion of errors in Part II.)

From this directory you may obtain the names of the files you wish to access for repair in the File Zap function (see the next section). If the directory consists of more than one page the system will prompt "There's more - press any key" and await entry of any key except <BREAK> to display the remaining pages. To exit to the Master Menu from any but the last directory page press <BREAK>. When you reach the last directory page the system will prompt with "Last page, hit any key." Press any key to exit to the Master Menu.

File and Disk Access

The following sections cover accessing a diskfile or the disk as a whole to make repairs. The first command, Verify, lets you determine which sectors are faulty, and what the errors are. The Read and Write Sectors commands together allow you to copy your disk by reading many sectors into memory and writing them to whatever disk or portion of the disk you desire. File Zap lets you access particular files for repair; Disk Zap lets you access the disk as a whole for repair.

<V> Verify Disk

When you encounter a disk error when using one of your disks which prohibits disk access, you will want to locate the error and find out its nature. The Verify Disk command performs this function. Upon pressing <V> from the Master Menu the VIP Disk-ZAP searches through the entire disk, through each sector of each track, looking for errors. When it finds an error it attempts several times to read the faulty sector, and then stops so that you may note the location (drive, track and sector) of the error. It also gives a description of the error on the COMMAND LINE at the bottom of the screen (see Part II) and prompts "SKIP SECTOR OR EXIT?" Do not be alarmed when, as the system tries to read the same sector, it changes the contents of the sector. This indicates that it has encountered a faulty sector and cannot read it correctly.

After you have noted the location and nature of the error, press <S> to skip the faulty sector and continue to verify until it encounters the next error, if any. (If you press <BREAK> you will exit the Verify Disk function and return to the Master Menu.) Continue with this process until the entire disk has been verified. The VIP Disk-ZAP allows you to read sectors on tracks beyond the standard 35th track of the Color Computer operating system for systems which have an extended

35-plus track format. When the system reaches the end of the 35th track it will prompt: "Continue or exit?" If you have a 35-plus track format, press <C> to continue; if you have the standard 35 track format, press <E> to exit. Once you have located the errors and found out their sources, you may then proceed to repair the disk through either the File Zap or Disk Zap menu commands.

<R>, <W> Read Sectors/Write Sectors

These two commands are a matched pair used to copy or manually backup portions of your disk to another location, either another disk or another portion of the same disk. This copy function is useful if you wish to copy your entire disk to another diskette, if you wish to save a copy of your directory track before you operate on the original, or for any other of your copy needs.

To begin, you must Read the sectors to be copied into computer memory. A 32K computer will hold 92 sectors (5 tracks, 2 sectors); a 16K will hold considerably less, and a 64K will hold considerably more. To Read the sectors to be copied press <R> from the Master Menu. The system will prompt at the bottom of the screen "Drive, Track, Sector?" and await your entry of the starting point of the sectors to be read. Recall that the drive, track and sector numbers may be input either in the hexidecimal or decimal number system. The base used is toggled with the Toggle Base command by pressing <SHIFT><@>; you may also force input of either base while in the other base (see discussion of Toggle Base above).

After you have entered the drive, track and sector numbers and pressed <ENTER>, the system will prompt "Sector Count?" The system will then await the number of sectors you desire to be read into memory. Below this prompt will be a reminder, in the base you have

toggled, of the maximum number of sectors which will fit in your computer buffer, reading "Count Limit:(#)." When you have input the number of sectors to be read into memory, beginning with the specified sector, and pressed <ENTER>, the system will read the designated sectors.

NOTE: The VIP Disk-ZAP allows you to read sectors on tracks above the standard 35th track of the Color Computer operating system for systems with extended 35-plus track formats.

Once these sectors are in memory the system will prompt "Request?" You then may Write the sectors to a new location. The procedure is the same as for the Read command, including the prompts, except that it is initiated by pressing <W>, and after entry of the number of sectors to be written, the system will prompt with "Do you really wish to write?" A <Y> response will cause the sectors to be written to the designated location; any other response will abort the Write. All writes are verified after being written to assure accurate transfer of data.

NOTE: that the Read Sectors command in the Master Menu is not the same as the Read Sector command in the File Zap menu. The latter command only allows one sector, the current one, to be read into memory. See the relevant section under File Zap.

<F> File Zap

When you press <F> for File Zep from the Master Menu the system clears the Master Menu and sets out the following prompts:

DRV: 00 TRK> 00 SEC: 00 UXDEC ASC

File Name: <Flashing Cursor>

File Name: (Last filename used)/(last extension, initially VIP):(last drive number)

The system awaits your entry of the name of the file which you wish to inspect or repair. You must enter the filename, the extension used and the drive in which the disk containing the file is located. The extension is necessary when you are changing extensions; thus, if you are accessing an ASCII file with the extension "BAS" right after loading the program you must include the extension BAS or the system will apply the default extension "VIP". The correct name and extension may be obtained by using the directory command (see section <A> Directory above). The filename must be separated from the extension by a slash ("/") or a period ("."); the drive number must follow the extension and must be preceded by a colon.

EXAMPLE: File Name: TESTFILE/VIP:0

The above example responds to the prompt with the filename "TESTFILE", with the extension "VIP", located in Drive 0.

The VIP Disk-ZAP stores the name of the last diskfile accessed in a buffer to eliminate the need to repeatedly type in the same filename. Below the filename request line is a listing of the last accessed filename. If you desire to access the same file you may press <ENTER> instead of retyping the filename.

Upon entry of the filename, or pressing of <ENTER>, the system will attempt to access that file. If the file is not found, a prompt "FILE NOT FOUND" will appear on the command line. Perhaps you forgot the extension? Try again. If the file is found, it will be accessed, on the command line will appear the prompt "FILE FOUND", and the screen will change to display the File Zap menu giving the commands which may be used to access, modify and transmit the file. The menu lists:

File Name <N>: Gives the name of the last file accessed.

Directory <A>: Supplies a directory of the disk in the drive specified.

Last Granule <E>: Accesses and displays the first sector of the last granule of the named file.

First Granule <F>: Accesses and displays the first sector of the first granule of the named file.

Read Sector <R>: Commands the system to read the current sector.

Modify Sector <M>: Enters the modify mode to modify the sector displayed on the screen.

Print Sector <P>: Allows the current sector to be sent to a printer or any other RS-232 device.

Set Baud Rate <B>: Elicits a menu for selection of the proper baud rate for transmission of data.

Minus 1 Sector <LEFT ARROW>: Moves back one sector in the diskfile.

Add 1 Sector <RIGHT ARROW>: Moves ahead one sector in the diskfile.

Minus 1 Gran <UP ARROW>: Moves back a single granule in the diskfile.

Add 1 Gran <DOWN ARROW>: Moves ahead a single granule in the diskfile.

Exit File Zap <BREAK>: Exits from File Zap to the Master Menu.

Below the menu is a listing of drive, track and sector numbers, listing the numbers of the last file accessed (in the base selected), the hex or decimal and ASCII flags and the prompt "File Request?" asking for a command from the File Zap menu.

The commands in the menu will be discussed in a logical order below. The first set of related commands are those used to access the file. These include: 1) the File Name and Directory commands which allow you to call up a particular diskfile and to keep track of which file you are working on; 2) the First and Last Granule commands used for easy movement to the beginning or end of the file called up; 3) the Read command used to peruse your file sequentially; and 4) the arrow commands which allow sequential access back and forward of sectors and tracks when accessing a file.

The second set of commands allow manipulation of the diskfile to be repaired. These commands are all used in the Modify function, and include the ASCII/HEX input option, the arrow keys used to move the cursor about the sector being modified, and the commands used to save the zapped sector to the disk. The third set of commands relate to Printing or transmitting a sector. This set includes the Print and Set Baud Rate commands. The final function is controlled by the <BREAK> command which allows exit from the File Zap function.

File Access Commands

The Name and Directory commands allow you to call up the file which requires repair. The Read command allows you to peruse the file. The First and Last Granule commands provide an easy means to get to a convenient place to begin inspecting a file. Once the file has been displayed, you may use the arrow keys to move back and ahead in the file.

The Screen Display

Before going on to a discussion of how to access a file, an explanation of the screen display of an accessed file is in order. When the VIP Disk-ZAP accesses a file it calls in one sector of the file at a time for display. The screen is split to display the sector being accessed. The top half of the screen contains an ASCII representation of the entire sector. The bottom half contains the usual indicators and flags. Provided the sector contains an ASCII file and you have retained the ASCII option with the TOGGLE X-LATE option from the Master Menu, the sector will appear just as it did when you created it - in its ASCII format with upper and lowercase letters, etc.

Many of you are experienced programmers and may not be used to seeing a pure ASCII display; instead, you are used to seeing the display presented by your monitor program. To accomodate such users we have included an option to allow them to select to see the accustomed non-ASCII display. To select that option press <X> to Toggle X-Late (see the discussion under Getting Started). When <X> is pressed from the Master Menu, the "UXDEC" flag will change to "U DEC" and the heretofore readable sector displayed will change to skads of gobbledygook - punctuation marks, parentheses, numbers and uppercase characters. These are all representations of the same sector, but using a different system than the standard ASCII display used in the ASCII option.

Although many of you will never choose other than the ASCII option, you should be aware that if you accidentally press <X> while in the Master Menu, you will get a very different screen display of the sector accessed. The following discussion of the ASCII system is included to help you understand the reason for this different display and to help you understand how to read and change the ASCII or non-ASCII display when it becomes necessary to repair or modify your disk.

The ASCII System

ASCII is a standard for symbols used in communications. "ASCII" is an acronym for "American Standard Code for Information Interchange". The ASCII standard potentially contains 256 symbols which are represented by numeric values from 0 to 255 decimal (0 to FF hexidecimal). The first 128 ASCII symbols constitute the Standard ASCII Character Set and are absolute standards for communication symbols, such as the letters of the alphabet, punctuation and control codes. The final 128 symbols (from 128 to 255) constitute the Extended ASCII Character Set, which will be discussed later.

Before going on with this discussion you should become familiar with the Standard ASCII Character Set chart in the Appendix. The standard ASCII character set is comprised of the 128 ASCII symbols from decimal 0 through 127. The first column gives the first 128 ASCII symbols in order; the second column gives the display of those symbols using the non-ASCII option; the third column gives the display of those symbols using the ASCII option; the fourth column gives the decimal numeric equivalent of the ASCII symbol; column five gives the hexidecimal equivalent; and the final column explains how to generate the ASCII symbol and its screen display equivalent through your Color Computer keyboard while modifying a sector using the ASCII input option. Thus, from the Appendix you can see that to generate the ASCII symbol capital "A", and its screen display capital "A" (in either the ASCII or non-ASCII option) you either place the hex number 41 in the appropriate location in the hex display in the hex option or, in the ASCII option, input a capital "A" from the keyboard in the proper location in the ASCII display.

The first 32 ASCII symbols, with numeric values from 0 to 31 decimal, are control characters. They were specifically devised to be used to control functions of

devices which accept ASCII data. The next 96 ASCII symbols, from 32 to 127 decimal, are the alphabet, in upper and lower case form, the numbers, and the conventional symbols seen on the typewriter keyboard such as the colon, the ampersand, the dollar sign, etc. In every system adhering to ASCII the ASCII symbols represented by the numbers from 32 to 127 decimal will be the same.

The Extended ASCII symbols from decimal 128 to 255 are different. There is no standard symbolic equivalent for them. Each system (computer, printer, etc.) may use these decimal numeric equivalents to produce different symbols. Because of this lack of standard ASCII symbols, monitor display for the numeric equivalents from decimal 128 through 255 are not uniform; every system assigns different symbols to the numbers from 128 to 255. Therefore the Appendix does not show anything for those decimal equivalents. The Color Computer represents these numeric equivalents as graphics blocks of various colors (see your BASIC manuals). Cards are available commercially listing the numeric equivalents and the corresponding pixel displayed.

The VIP Disk-ZAP is totally compatible with ASCII. This means that all functions of this program requiring modification of a disk allow modification in the ASCII mode, and the memory contents holding the diskfile or sector are in the ASCII format. The screen display in the ASCII option is also ASCII compatible. The non-ASCII option display, however, IS NOT! If you call up a file using the non-ASCII option, you will immediately see that the screen display is not ASCII compatible. What you put in your file as normal looking words with real letters have become real gobbledygook, the letters being replaced by numbers, punctuation marks and whatever odd doohickey. This aberration is due to the construction of your Color Computer, since the VIP Disk-Zap non-ASCII option merely uses the display that the Color Computer generates. As you can see, the character generation component of the Color Computer

which displays the characters on the screen is not ASCII compatible.

Let me explain this further. As was said, ASCII is just a set of numbers assigned to some communication symbols so that manufacturers of terminals, printers, etc., could create uniform products. There is no particular reason the number 97 decimal (61 hex) has to be assigned to the letter "a"; it was just the numeric equivalent chosen for the ASCII system. Every system adhering to ASCII will have the number 97 decimal equal the letter "a".

Why all this concern about numbers? Because your computer stores information in numeric form instead of as characters. When you press a key on the keyboard, what you are really doing is putting a number into memory, that number being the numeric equivalent of the ASCII symbol you have generated. It just so happens that the letters of the alphabet and other standard symbols on your keyboard ARE the ASCII symbols themselves, and thus, when you press a key the corresponding numeric equivalent for that ASCII symbol is put in memory. In a totally ASCII compatible system, when you press the "a" key, you have generated the ASCII symbol "a". How is "a" represented in your memory? By looking in the Appendix you can see that the numeric equivalent for the ASCII symbol for "a" is decimal 97 (61 hex). (Actually the memory contains the binary equivalent of the decimal number 97. For convenience we will refer to decimal numbers when referring to memory contents.) When you press "a", your memory receives a decimal 97 in the appropriate memory location.

The monitor (your TV) which displays your memory is controlled by a device which generates characters from the ASCII numeric equivalents in the memory. If the character generator is ASCII compatible, the decimal 97 from the memory, generated when you pressed the "a" key, will go to the character generator, which in turn will tell the monitor to show an "a" on the screen. But what

if the character generator is not ASCII compatible? This means that it may assign a different symbol to the ASCII numeric equivalent.

This is exactly what causes the wierd screen display of your ASCII files in the VIP Disk-ZAP when using the non-ASCII display option. The Color Computer's character generator assigns different symbols to many of the ASCII numeric equivalents. Most notably, the ASCII symbols for the lower case alphabet have been exchanged, in the character generator, with the control symbols CTRL A to CTRL Z with their screen display. The Appendix shows this. As was shown, in an ASCII compatible system, a decimal 97 will generate the letter "a". In the Color Computer screen display, however, decimal 97 generates an inverse exclamation point. By glancing at the screen display of the Color Computer for the ASCII lower case alphabet (from decimal 97 thru 122) you can understand why your wonderful file has been turned into garbage in the non-ASCII display option.

This display difference should have absolutely no effect on you. You can choose to use either the pure ASCII display or the Color Computer non-ASCII display, whichever you are most used to or like the best. The information on numeric equivalents of ASCII symbols will prove very useful once you begin making modifications to your files. This concept is crucial to proper Zapping.

Now that you understand the display choices and the ASCII system, we can return to the discussion of the File Access commands in the File Zap menu.

<N> Name

This command was described above in explaining the prompting resulting from pressing <F> to enter the File Zap mode. See that discussion. The same command is also available from the Master Menu. Note that the Disk Zap mode lacks this command since that mode deals with

the disk as a whole instead of with individual files on the disk.

<A> Directory

This command is also identical in function and execution to that described for the same command in the Master Menu. The Directory command is also not available in the Disk Zap mode, as that mode does not call up individual files listed in the disk directory.

<E> Last Granule

This command is used to access the first sector of the last (or only) granule of the designated disk file. When it is selected, that sector will be displayed on the screen, and you will be prompted with "FILE REQUEST:". You may move through the file for inspection using the arrow keys (see discussion below). You may press <ENTER> to return to the File Zap menu, <BREAK> to return to the Master Menu, or any File Zap command key to execute a command.

Whenever a sector is actually accessed from the File Access or Disk Access commands, along with the other indicators for drive, track, sector, ASCII, number base and upper or lower case will appear a GRAN indicator. This indicator tells you the granule number of the sector accessed in the number base you have selected from the Master Menu. This number will prove useful when it becomes necessary to repair a disk (see the section on Disk Structure in Part II).

<F> First Granule

This command is used to access the first sector of the first (or only) granule of the designated disk file. When it is selected, that sector will be displayed on

the screen, and you will be prompted with "File Request:". You may move through the file for inspection using the arrow keys (see discussion below). You may press <ENTER> to return to the File Zap menu, <BREAK> to return to the Master Menu, or any File Zap command key to execute a command.

<R> Read Sector

This command is used to load the current sector of a designated disk file, i.e., the last one accessed, into memory and display it on the screen for inspection. When it is selected, that sector will be displayed on the screen, and you will be prompted with "FILE REQUEST:". You may move through the file for inspection using the arrow keys (see discussion below). You may press <ENTER> to return to the File Zap menu, <BREAK> to return to the Master Menu, or any File Zap command key to execute a command.

Left/Right Arrow; Up/Down Arrow

In normal file access functions, used to locate the sector to be zapped, the arrow keys move you through sectors and granules of data. The left and right arrow keys move you back or ahead one sector in the file - thus the menu descriptions "Minus 1 Sector" and "Add 1 Sector". The up and down arrows move you back or ahead one granule in the file - thus the menu descriptions "Minus 1 Gran" and "Add 1 Gran". In using any of these arrow commands, if you reach the boundaries of the file a prompt will appear at the bottom of the screen telling you that you have reached "First sector of file", "First Granule of file", "Last sector of file" or "Last granule of file".

NOTE that the arrow keys perform different functions when in the Modify function.

<M> Modify Sector

The modify function is the heart of VIP Disk-ZAP. It is primarily with this function that you will repair the disk - or ZAP it (see Part II).

Modification is done one sector at a time. The first task before modification is to determine what modification is necessary to make a repair by using the Verify command or by inspecting the diskfiles or the disk as a whole. Once you have determined what sectors need modification, you then must locate the sector using the Read, First and Last Granule commands and the arrow key commands. The Modify command will access the first sector of your diskfile or the last sector accessed for Zapping.

When you locate the sector to be modified, on the Command Line at the bottom of the screen will be the prompt "File Request". Press <M> to begin modification. The screen will go blank, and then a new screen display will appear (the screen display in the modify function is described below). The cursor rests at the first byte of the sector (relative byte zero), awaiting your action.

To return back to the File Zap menu press <BREAK> and then <E> for Exit. Press <BREAK> twice to return to the Master Menu.

The Screen Display

When the modify function is called, the screen display is split into two parts. On the top half of the screen is the ASCII display of the sector of the disk to be repaired. The display contains one sector from the disk, displayed in eight lines of text, each containing 32 bytes of data, for the total 256 bytes per sector. The bottom half of the screen contains a hex display, the Gran, Drive, Track and Sector indicators, the UXDEC

and ASCII flags and a command line for prompts from the system and input from the user. The hex display is comprised of a relative byte counter and four lines of eight bytes each of numbers which correspond to the ASCII display characters. The 32 bytes of the hex display are equivalent to the single sector ASCII display (i.e., 1/8 sector); thus the eight lines of the ASCII display will be represented by eight consecutive hex displays.

A double cursor indicates the current position in both the displays. The hex number indicated by the cursor in the hex display is the numeric equivalent for the ASCII symbol at the position of the cursor in the ASCII display (see the discussion of the ASCII system above). The relative byte counter indicates the byte number, in the base chosen from the Master Menu with the Toggle Base command, at the position of the cursor. This helps you keep track of where you are in the split display, especially in the eight consecutive 32 byte hex displays per sector.

The hex display provides the numeric equivalent, in hexidecimal, of the ASCII symbol at the position of the corresponding cursor in the ASCII display. Numeric equivalents of ASCII symbols are given in the Appendix. Thus, the symbol "A" in the ASCII display will correspond to the number 41 hexidecimal in the hex display. (Note that the screen display may differ from the ASCII symbol that the numeric equivalent represents when the non-ASCII option is chosen - see the discussion of the screen display above.)

Movement in the Display -

Movement through the ASCII and hex displays is done via the arrow keys. The arrow commands have a different effect in the modify function than in other file access functions (see discussion above). When in the modify function the arrow keys are used instead to move you

about within the sector. The left and right arrows now move the cursor one byte back or forward in the sector. The shift key plus the up or down arrow now moves the cursor left or right eight bytes in the ASCII display and simultaneously up or down one line (eight bytes) in the hex display. The up and down arrow keys alone move the cursor up or down one line in the ASCII display (32 bytes) and simultaneously up or down four lines (one 32 byte display block) in the hex display.

Zapping the Sector

Changes to the sector may be made through alteration of the altered sector displayed and writing of the sector back onto the disk. Alteration may be done either by inputting numbers into the hex display at the position of the cursor or by inputting ASCII symbols into the ASCII display. The input options are toggled by pressing <CLEAR>. When the ASCII option is selected, both the number flag and the ASC flag will be displayed on the screen; when the hex option is chosen, only the number flag will be displayed.

When in the hel input option, you may ONLY input in the hexidecimal number system. Unlike when entering the drive, track and sector numbers (see the discussion of Toggle Base above), input into the hex display may not be done in decimal, but only in hex. Therefore, even if you have elected to be in the decimal base by pressing <SHIFT><@> to toggle the number input from hex to decimal from the Master Menu, the system will interpret all numbers input as hexidecimal. The proper hexidecimal number to bg entered is chosen by looking at the Appendix and determining the ASCII equivalent you desire to generate, or by otherwise detevmining the number which will affect the desired result to the operation codes of the disk sector such as BASIC line pointers (see generally Part II on disk repair). When a number change is made to the hex display, the corresponding ASCII symbol will be altered at the

position of the cursor in the ASCII display. If an operation code is being generated, its corresponding screen display symbol will be displayed in the ASCII display as well (if the ASCII numeric equivalent is above 127 decimal, a graphics symbol will be displayed - see you BASIC manual).

When selecting the ASCII option, you may directly enter ASCII symbols into the ASCII display at the position of the cursor by pressing the key indicated by the fifth column of the Standard ASCII Character Set in the Appendix. When the ASCII symbol is generated, the symbol's hexidecimal numeric equivalent (see Appendix) will be displayed at the corresponding cursor position in the hex display. Note that some symbols may only be generated by selecting the number option and entering the numeric equivalent into the hex display. This is because the keyboard is not able to generate these characters. These instances are indicated by the phrase Hex Input Only in the fifth column of the Appendix.

When inputting data, you may repeat the last byte entered by pressing <ENTER>. This is handy for filling or clearing sectors.

Writing the Zapped Sector Back to Disk

Once you have made the changes necessary to fix the disk, you may write the sector back to the disk. To write, press the <BREAK> key. On the command line will appear the prompt: "Write or return?" By pressing <R> you will exit the write command and return to the pre-write screen display in the Modify mode. If you press <W> the system will prompt "Drive, track, sector?" awaiting input of the location to which you wish to write the sector. If you press <BREAK> again you will exit back to the File Zap menu.

You may input in decimal or hexidecimal and you may specify any drive, track and sector. Once you have

input the numbers and pressed <ENTER> the system will prompt "Do you really wish to write?" Press <Y> to write the sector; press any other key to abort the write and return to the modify function with the same sector. Whenever the system writes to the disk, it verifies the write to make sure no errors were made in the transfer.

Once you have written a sector to the disk you may return to the File Zap menu, or back to the Master Menu to call up another sector for modification.

Printing Functions

The VIP Disk-ZAP provides the opportunity to print the contents of your diskfile or transfer the contents to another RS-232 compatible device. This is referred to as "Printing" a sector, and may be done both from the File Zap and the Disk Zap menus. This can be very useful for obtaining a hard copy of corrections or data you have entered into a disk for inspection, as well as for transferring a file to another system for future manipulation, such as by using the VIP Terminal.

<P> Print Sector

Printing a sector refers both to sending a sector to a printer and to transmitting a sector to any other RS-232 compatible device. The data will be sent to the printer or modem at a default baud rate of 600 baud; you may select a different baud rate by using the Set Baud Rate function accessible from the File Zap menu. If a baud rate different from the default rate is to be selected, it must be done before selecting the Print Sector function.

Before you may select the Print Sector option you must have a sector file resident in memory. When you are ready to Print, press <P>. When you press the <P> key the command line will give the prompt "Output to printer or modem?" If you press <P> in response to this

prompt, the system will present prompts for sending the sector to the printer. If you press <M> the system will prompt you to send the sector to a modem or other RS-232 device.

If you pressed <P> a second time to send the sector to a printer, the system will allow you to choose between sending a forty or an eighty character line. The system prompts: "40 or 80 characters per line," and awaits your input of a <4> or an <8>. The display will be sent to the printer when you input either of these. Your selection depends on the capabilities of your printer (see your printer manual). The printer will print the hex display on the left side of the paper followed by the ASCII equivalents on the right side of the page. Non-printable ASCII graphics or control code symbols are printed as periods. You may pause printing at any time by holding the space bar. Resume printing by pressing any key except <BREAK>. Pressing <BREAK> at any time will end printing.

If you pressed <M> to send the sector to a modem or other RS-232 compatible device, you will be given two options. Your first option is between sending the sector in its binary object code or sending it in its hex-ASCII format. The system will prompt with "Binary or hex ASCII output," to which you may respond with a <B> for binary or <H> for hex-ASCII. The binary form is that which would control the computer or is a tokenized form of a BASIC program; the hex ASCII form is that which could be used to inspect the contents of the sector.

If you choose to send a binary file, once you have pressed <B> the file will be sent. If you have chosen the hex-ASCII format, the system will again allow you to choose whether you desire the hex output in 40 or 80 character lines. To choose either, press <4> or <8> after the prompt "40 or 80 characters per line." When you press your selection the system will send the sector to the modem.

<B> Set Baud Rate

This function allows you to select the proper baud rate to use while using the Print Sector function. The default baud rate is 600; this feature need only be accessed if you desire a different baud rate. The VIP Disk-ZAP allows you to choose six different baud rates: 110, 300, 600, 1200, 2400 and 4800. When you press <B> a menu will appear showing the different rates and the corresponding key to press, from <1> to <6>. After you have selected the desired baud rate by pressing the appropriate key (<ENTER> will select 600 baud), the system will return to the File Zap menu for further work.

<D> Disk ZAP

The alternative method for disk repair, other than the File Zap function, is the Disk Zap function. This alternative allows you to view the disk as a whole and make repairs throughout the disk. This alternative is particularly helpful for making repairs to the directory, which may not be called up as a diskfile using File Zap, or for repairing a crashed directory or other sector. Disk Zap is selected from the Master Menu by pressing <D>. Once selected the Disk Zap menu appears awaiting your selection of the appropriate command. The Disk Zap menu has the following commands:

Last Track <E>: Accesses and displays the last track of the disk.

First Track <F>: Accesses and displays the first track of the disk.

Modify Sector <M>: Enters the modify mode to modify the sector displayed on the screen.

Enter Drive <J>: Allows you to specify a particular Drive, Track and Sector for manipulation.

Enter Track <T>: Allows you to specify a particular Track and Sector on the existing Drive for manipulation.

Locate <L>: Will allow you to find all occurrences of an ASCII or numeric string on your disk.

Print Sector <P>: Allows the current sector to be sent to any RS-232 device.

Set Baud Rate <B>: Elicits a menu for selection of the proper baud rate for transmission.

Minus 1 Sector <Left Arrow>: Moves back one sector in the disk.

Add 1 Sector <Right Arrow>: Moves ahead one sector in the disk.

Minus 1 Track <Up Arrow>: Moves back a single track in the disk.

Add 1 Track <Down Arrow>: Moves ahead one track in the disk.

Exit Disk Zap <BREAK>: Exits from Disk Zap to the Master Menu.

Below the menu is a listing of drive, track and sector numbers, listing the numbers of the last sector accessed, the UXDEC and ASCII indicators and the prompt "Disk Request?" asking for a command from the Disk Zap menu.

Many of the commands in the menu are the same as those available from the File Zap menu. This will be noted as each is discussed below. The commands in the menu will be discussed in a logical order below.

The first set of related commands are those used to access the disk. These include: 1) the Enter Drive and Enter Track commands which allow you to call up a particular drive, track and sector of a disk and to keep track of which track and sector you are working on; 2) the First and Last Track commands used for easy movement to the beginning or end of the disk; 3) the Locate command used to find all occurrences of a particular string sequentially in the disk; and 4) the arrow commands which allow sequential access backward and forward of sectors and tracks when accessing a sector.

The second set of commands allow manipulation of the disk to be repaired. These commands are all used in

the Modify function, and include the ASCII/hex input option, the arrow keys used to move the cursor about the sector being modified, and the Write command used to save the Zapped file to the disk.

The third set of commands relate to Printing or transmitting a sector. This set includes the Print and Set Baud Rate commands. The final function is controlled by the <BREAK> command which allows exit from the Disk Zap function.

Disk Access

Initial disk access is made by using the Enter Drive and Enter Track commands. For those with only one disk drive or with a desire to use only one disk drive, there will be no need to use the Enter Drive command. The First Granule and Last Granule commands as well as the Locate command will take you to a place to begin work. Once the area has been located, you may move easily about the disk from that point by using the First and Last Track commands and the Left and Right Arrow and Up and Down Arrow commands.

<T> Enter Track

This command is initiated by pressing <T>. Upon pressing <T> the prompt "Track, Sector?" appears below the menu on the command line. You may specify the track and sector in either the hexidecimal or the decimal number system. See the Toggle Base command in the discussion of the Master Menu. To access a particular track and sector, enter the desired numbers, separated by a comma, after the prompt and press <ENTER>. If you desire to access thellast track and sector called up, press <ENTER>; if you desire to access a different sector in the same track, you need only input a comma to indicate the same track and then the sector number.

EXAMPLE 1: TRACK, SECTOR? 5,6<ENTER>
EXAMPLE 2: TRACK, SECTOR? ,8<ENTER>

In the first example sector six of track five will be accessed. In the second example, sector eight of the last accessed track, here track five, will be accessed.

NOTE: It is only when entering the (drive+, track and sector number or when using the Locate command that the decimal number system may be used for input.

When you have entered the numbers of the track and sector you desire and pressed <ENTER>, the system will locate that sector and display it on the screen. By the ASCII display will be the GRAN, DRV, TRK and SEC indicators giving your present location, the UXDEC flag, and below that a prompt "Disk Request." You may enter another command from the Disk Zap menu or you may move from there by using the left, right, up and down arrows to move around the disk by sector and by track (see the discussion of the arrow key functions above in the File Zap section). Pressing <ENTER> at any time will exit to the Disk Zap menu; pressing <BREAK> will exit to the Master Menu.

<J> Enter Drive

This command is initiated by pressing <J>. Upon pressing <J> the prompt "Drive, Track, Sector?" will appear below the menu on the command line. You may specify the drive, track and sector in either the hexidecimal or the decimal number system.

As with the Enter Track command, you must enter the number of the desired drive, track and sector to be accessed, separated by commas, and press <ENTER>. Replacing the drive or track number with the comma will specify that the same drive and track number as last accessed is desired. To access the identical drive, track and sector as last accessed press <ENTER>. When

you have entered the numbers of the drive, track and sector you desire, the system will locate that sector and display it on the screen. By the ASCII display will be the GRAN, DRV, TRK and SEC indicators giving your present location, the UXDEC flag, and below that a prompt "Disk Request." You may enter another command from the Disk Zap menu or you may move from there by using the arrows keys to move around by sector and by track. Pressing <ENTER> at any time will exit to the Disk Zap menu; pressing <BREAK> will exit you to the Master Menu.

<E> Last Track

This command will cause the system to call up the first sector of the last track of the disk. Press <E> for execution. You may use the Locate and arrow commands to continue from there. Press <ENTER> to return to the Disk Zap menu.

<F> First Track

This command will cause the system to call up the first sector of the first track on the disk. Press <F> for execution. You may use the Locate and arrow commands to continue from there. Press <ENTER> to return to the Disk Zap menu.

Arrow Commands

These commands are the same as those in the File Zap function. Refer to that section for a discussion of them.

<L> Locate

This command can be of great use to those who wish to use the VIP Disk-ZAP to correct just one or two mistakes in an assembly language source program, or find any occurrence of an ASCII string or a number in a disk for that matter. When you press <L> to execute this command, the prompt "Locate:" appears on the command line below the menu awaiting your input. Input may be either in numbers or ASCII, toggled by pressing <CLEAR>.

When locating numbers, the ASCII flag must be off (by pressing <CLEAR>), and if a string of numbers is being located, each number in the string to be located must be separated by a comma. For example, if you wished to locate a string with two numbers, 88 and 135, you would enter 88,135<ENTER>.

ASCII strings may be input in uppercase only, or in both upper and lower case. Case is originally uppercase only and is toggled by pressing <SHIFT> <0>. Note that when you are in the lowercase mode using the non-ASCII display option (see discussion of screen display above), lowercase characters will be represented by punctuation and other anomolous symbols. This is due to the non-standard character generator chip used by Tandy in the Color Computer.

After you have entered the desired ASCII or number string, press <ENTER>. The system will locate the first occurrence of the string in the disk, display that sector in the ASCII display and will stop there with the cursor immediately following the string. Below the ASCII display is a BYTE indicator which indicates the relative location of the last character in the string in number of bytes, in the number base you selected, from the beginning of the sector.

On the command line will appear the prompt "Continue or Exit?" To continue to the next occurrence of the string press <C>. You may continue to the end of

the disk and beyond. The VIP Disk-ZAP supports certain systems which use tracks beyond the 35th track accessible by the standard Color Computer operating system. When the program reaches the end of track 35 it prompts: "Continue or Exit." Those of you having a system using a format of over 35 tracks may continue Locating through those files by pressing <C>. Those with the standard 35 track format must press <E> to exit the Locate function, When you exit the Locate function the ASCII display will remain intact and you will be prompted: "Disk Request" on the command line. You may then proceed to call up any command from the Disk Zap menu, such as Modify (if you want the Disk Zap menu press <ENTER>), or press <BREAK> to exit to the Master Menu.

During a Locate, or any read function for that matter, the system may encounter a faulty sector. If so it will perform a Verify function on the sector to determine the nature of the error. For more information see the Verify section above.

Disk Repair Commands

This section consists of the Modify command and its sub-commands. The Modify command and its sub-commands in the Disk Zap function are identical to those in the File Zap function. Please refer to that section for a thorough discussion.

Print Commands

The print commands consist of the <P> Print Sector command and the <B> Set Baud Rate command. These commands in the Disk Zap function are identical to those in the File Zap function. See that section for a thorough discussion.

Killing Disk Files

<K> Kill

This command is used directly from the Master Menu. It is used to kill diskfiles. When you press <K> a prompt "Kill Disk File?" will appear on the command line below the Master Menu. You must then input the name of the diskfile that you desire to kill, including extension and drive number, and press <ENTER>. The system will not automatically kill the file, but will first seek authorization by prompting "Do you really wish to kill?" Pressing <Y> will kill the file; pressing any other key will abort the Kill and bring back the "Request?" prompt for further action from the Master Menu.

PART II: ZAPPING TECHNIQUES

Introduction

With care - extreme care - and luck you should never have a need for the VIP Disk-ZAP. All you have to do is always backup the files you are working on, both periodically while you are working on them and when you are finally done. You also must take special care of your entire computer system and your disks. You can keep the bus connections on your disk pack plugs clean using a large pencil eraser; you keep your disks in shape by not using them for second base in a whiffle ball game, or a swab for your ketchup. By the way, data is stored on the back side of the disk, so avoid touching all exposed parts of the disk, especially those on the back side.

Still, even with the utmost of care, your disks can become garbled. Your wife could put them into a toaster, or, more likely, the operating system could be finicky and deconstruct both your master and backup disks. Then it's VIP Disk-ZAP time. One word of warning, though. Not every crashed disk is repairable; some crashes may be caused by defective or damaged disks.

There are generally three types of persons who will use the VIP Disk-ZAP. The first is the experienced machine language programmer. Such a person will know most all there is to know about zapping disks, and will probably never even read this section.

The second type of person is the intermediate to experienced BASIC or other high level language programmer. This person is constantly using the disk system to write programs and is likely to encounter disk errors. Since BASIC programs are normally saved to the disk in a tokenized machine-language form, correcting such files may be difficult without learning some

machine language concepts. This manual cannot hope to teach those concepts, and instead is devoted to helping those who save files in ASCII. (We have, nevertheless, provided the curious with a list of the BASIC tokens in the appendix. Gook luck!)

BASIC programmers can save a lot of trouble if, instead of saving programs in the usual manner, they save the programs in the ASCII format (SAVE"filename",A<ENTER> - see your BASIC and Disk manuals). You can then make full use of the VIP Disk-ZAP without having to learn machine language. Added to this is the fact that you can then load the program into an editor, such as the VIP Writer, for fast and easy editing.

The third type of person likely to use this program is the individual who is not and never wants to be any kind of programmer. He or she just wants to fix disks to retrieve valuable ASCII files created using some utility program such as a word processor, like the the VIP Writer. Since recovery and restoration of ASCII files is the true purpose of this program, this portion of the manual has been written with this person in mind. Although we attempt in this part to teach some rudimentary concepts of disk repair, we do not promise to teach you any but a minimal amount on the subject. To become an expert you must read reference works and get plenty of experience.

Disk Structure

In order to understand disk structure, you must understand how numbers are used in many computer applications. Often, items in computer applications are numbered beginning with zero instead of one. Thus, there are 35 tracks to the disk, numbered from 0 to 34. A convention has been developed to refer to such a counting method which considers 0 as the first number. These numbers are referred to as "relative" numbers.

Thus, the first track is relative track zero; the 35th and last track is relative track 34. This convention can be very confusing, so be careful to keep tabs on which is being referred to.

Before beginning to repair a disk you must first understand how data is stored on a disk. Chapter 11 of your Color Computer Disk System book is devoted to this topic and is essential reading. Since Western Digital Corporation is the maker of the Disk Controller chip which controls your disk drive, you may also turn to their publications on the controller chip for some very technical explanations. See the following of their publications: FD 179X-02 Floppy Disk Formatter/Controller Family and FD179X Application Notes.

In discussing disk structure, we've got to be concerned about five categories of disk structure. From smallest to largest unit these are: Bytes, Sectors, Granules, Tracks and the Directory.

First off, A byte is a unit of data equivalent to one character, such as the letter "Z" or a comma. The data you putcon your disk is put there in byte units, and the remaining parts of the disk are made up of conglomerations of bytes. Of course, a character is not put on the disk; instead, the numberic equivalent of a character is put there. If you are not storing ASCII characters, numbers are still stored, but they don't equate to characters, but to some token or operation code. The numbers cre storee in binary form, that is, ones and zeros. Since those are hard for us to read, they are converted into hexidecimal (and sometimes decimal) numbers for monitor display. Each byte contains a combination of eight ones and zeros, allowing representation of numbers of up to FF hex (relative 255 decimal). Each one or zero is a bit, so there are eight bits to the byte.

A Sector is the next smallest part of disk structure. With the Radio Shack Color Computer operating system, each sector contains 256 data bytes. It is these data bytes you see in the ASCII display when using this program. The sector is also surrounded by identification bytes used for such things as telling the system where this sector is for reading from and writing to this sector. These ID bytes, which are set when the disk is initially formatted, cannot be displayed on the screen, nor can they be zapped using this program. In these ID bytes are also bytes used to make sure what was being written to the sector got there intact. These bytes are called cyclical redundancy check ("CRC") bytes. If the sum of these bytes is not what the computer expects it to be, i.e., the "checksum" is off, a CRC ERROR occurs. More on this later.

NOTE that since the number of data bytes per sector is controlled by the operating system, some operating systems have sectors with fewer or more than 256 data bytes.

The next largest unit of disk structure is the Granule, but since it is more easily understood after you understand what a Track is, the Track will be discussed first. With the standard Color Computer operating system a disk contains 35 tracks numbered from 0 to 34, and each track contains 18 sectors, numbered from 1 (not 0) to 18. Each track is surrounded by extra bytes for ease of location by the disk drive. (Some systems allow use of formats having more than 35 tracks per disk, and those systems are also supported by the VIP Disk-ZAP.) Of the 35 tracks, 34 are available for your files. The other track, track 17, is reserved for the directory (discussed below).

Now to the Granule. A granule is not so much an element of disk structure as it is a function of the operating system. It is the smallest unit in which files may be saved to the disk by the operating system. In the Color Computer that size is nine sectors. That

is exactly one-half a track. Thus, with the 34 tracks available (the directory track is not available) for your files, there are 68 granules on the standard Color Computer disk. The VIP Disk-ZAP contains a GRAN indicator on the screen when any disk access is made to allow you to keep track of which gran you are in.

As was said above, Track 17 (11 hex) is set aside for system use for the Directory. It contains the information the operating system needs to catalog what your files are, where they are located on the disk, and how much space they occupy. The directory track will be the focus of much of your attention when repairing disks, so it will be given the most detailed attention here.

Like other tracks, the Directory Track contains 18 sectors, but only ten are presently used by the operating system. The directory can be divided into two sections: the file allocation table (more commonly and from now on called a granule allocation table - GAT) residing in sector 2; and the directory entry section, sectors 3 to 11, which are devoted to identification information about each of the files stored on the disk.

Although the GAT sector occurs first in the track, it will be more easy to understand if it is discussed after the directory entry section. Directory entries are each allocated 32 bytes, but only 16 are used. Before describing the functions of the various bytes, remember that when you are looking at the ASCII display you must go into the Modify function to see the numeric equivalent for that ASCII symbol. In the Modify function, you will be given the hexidecimal numeric equivalent of the ASCII symbol in the HEX display in the bottom half of the screen. To find the hexidecimal equivalent of any particular ASCII symbol, move the cursor over that symbol and look at the number at the position of the cursor in the HEX display.

The first eight bytes, starting from zero, are reserved for the filename. The next three bytes are for the extension. Note that the "/" used to separate the filename and the extension when you save a file to disk is not stored in the directory. That is because it is a command to the operating system to treat the next three bytes as an extension. The eleventh byte tells what type of file this is. The binary equivalent of 0 means it is a BASIC program; 1 is a BASIC data file; 2 is a machine language program and 3 is a text editor source file. The twelfth byte tells whether the file is stored in ASCII or binary format, 0 for binary and 255 (FF hex) for the ASCII format. The thirteenth byte tells the relative number of the first granule in the file. It may be from 0 to 67 (43 hex). Bytes 14 and 15 tell the relative number of bytes used in the last sector of the file. Since a sector in this operating system may contain only 256 bytes, byte 14 will always be 00 and byte 15 will give the end-of-file byte unless all 256 bytes are used, in which case byte 14 will be 01 and byte 15 will be 00. The remaining 16 bytes in the directory entry are reserved for future use. Each of the 16 bytes used will become of immense importance when you desire to reconstruct a directory or a file.

The Granule Allocation Table (GAT) sector contains 68 bytes of concern to us. Each of these bytes relates to each of the granules on your disk, in sequential ascending order, from 0 to 67 (43 hex). Thus, byte 0 corresponds to granule 0; byte 43 hex (67 decimal) corresponds to granule 67. (Remember that the directory track, track 17, is not used for data storage, and thus is not allocated granules.) The contents of these bytes corresponding to the granules may contain three different types of numbers (in hex): FF, 00 to 43 and C0 to C9. Look at the hex display of the sector in the Modify function to see the numeric value of each location. The BYTE indicator will tell you which byte (in hex) and thus which granule you are considering.

If the byte contains an FF hex (255 decimal), the corresponding granule is free and not part of any disk file. If the byte contains any of the numbers from 0 to 43 hex, this indicates that the corresponding granule is part of a disk file, and the number points to the next granule in the file. If the byte contains any of the numbers from C0 to C9 you are told two things. The C indicates that this granule is the last granule of a disk file. The number, from 0 to 9, indicates the number of sectors in the granule that are part of the disk file. Note that to determine the number of bytes in use in the last sector you may refer to bytes 14 and 15 of the directory entry for the file in the directory track.

This abstract explanation has probably floated by in several swigs of beer. Let's make it more concrete by tracing a hypothetical file named TESTNAME/VIP. When sector three of the directory track is called up, the name TESTNAMEVIP would occupy the first eleven bytes, from 0 to 10. TESTNAME would be the filename and VIP (VIP Writer file) would be the extension. Byte eleven would tell that this is a data file, byte twelve would indicate that it is an ASCII file. The thirteenth byte would tell you the relative number of the first granule of the file. Here let's assume the number in byte thirteen would be 5. With this number you could go to the GAT sector (track 17, sector 2), and you would then go into the Modify function. You would see both the ASCII and HEX displays. Recall that the number obtained from byte thirteen was 5. Paying attention to the byte indicator, you would move the cursor over the fifth byte. (It is most helpful for granule tracing to have selected the hex option from the Master Menu so that the BYTE indicator and the HEX display numbers will both be in hex.) In the fifth byte would be a number other than FF since the granule would be part of the TESTNAME disk file. Let's say the number would be 32 (20 hex). This number would tell you the next granule in your file. You would then again look in the GAT sector to see what granule 32 would contain. It would not hold FF hex

since again it would be part of the disk file. Let's say this time that the number in byte 32 would be C3 hex. This number would indicate that this granule would be the last granule of your file (the C) and that only three of the nine sectors in the granule would be part of the file. Now you would know the last sector of the disk file. To find out the number of bytes in the last sector which are part of your file you would have to return to sector three of the directory track where the directory entry is located and check bytes 15 and 16 (relative bytes 14 and 15). Bytes 15 and 16 give a number which is the number of bytes in the sector, up to 256 decimal, used by your file. You would use this number and go to the last sector of your file to find the end-of-file point. If only a portion of the sector were devoted to your file, the rest of the sector would be filled with garbage.

One last thing remains for proper handling of the granule information: granule to track conversion. This is essential to be able to determine from the granule number obtained from the directory which track your files are on since this program and the disk are track-oriented. First, convert the hex number you have obtained into decimal for easy manipulation (see decimal to hex conversion chart in Appendix). Divide the number by two. The result will be used to obtain the track number. If the number is less than 17 it is the track number; if the result is over 16, add one to the result (because the directory track occupies track 17), and this is the track number. If the remainder after the division is 0, go to the first sector of the track; if the remainder is 1, go to the tenth sector of the track. To go from track number to granule number reverse this procedure. If the track number is under 17, multiply by two to get the granule number. If you are concerned with sector 10 or beyond of that track, add one to the result to get the granule number. If the track number is 18 or more, add an additional two to get the granule number. Remember to convert the decimal number back to

hexidecimal if you need to (i.e., you have selected the hex number option).

Now you know just about all you ever wanted to about where your file is on the disk.

The Operating System

Your disk system is comprised of hardware and firmware. The hardware is the disk drive; the firmware is the conglomeration of chips in you controller pack. Central to these is the floppy disk formatter/controller chip supplied by Western Digital Corporation (see above). This chip controls how the disk drive will work, and signals if things aren't working right.

In the controller pack also is Tandy's Color Computer operating system program chip. The disk operating system is a program supplied by Tandy which controls how your files are saved to and read from the disk. It determines the diskstructure for its purposes, such as the number of tracks, sectors and data bytes, and it controls formatting and killing the disk. It determines how and where files are placed, and what to do if files cannot be read or written. Knowledge of the inner workings of the operating system is not necessary to repair your disk. Its idiosyncracies should, however, be understood for proper comprehension of error prompts and for comprehension of the difficulty of restoring killed files.

The Color Computer operating system is fairly simple and straightforward. It doesn't offer many of the features available with operating systems on more expensive computers; yet, this makes the operating system less difficult to understand and makes it less difficult to repair operating system generated errors. The disk controller hardware is supplied by Western Digital Corporation. The Western Digital controller chip gives specific indications of particular errors

when they occur. The Tandy operating system does not directly use these indications. Instead, it generalizes the error indications into general error messages. These operating system error prompts thus may tell you very little about what the real error is. Errors are discussed in more detail below and this idiosyncracy will be discussed more there.

The second significant idiosyncracy relates to the manner in which disk files are "KILLED". When the Color Computer operating system kills a file it places a 00 in the first byte of the directory entry, thereby eliminating the first letter of the file name. Otherwise the directory entry for the file is not altered. The real change is done to the granule allocation table sector. The operating system places an FF in the byte corresponding to the number of each granule allocated to the file to tell the system that this granule is available for use. (The actual data in the respective granule on the disk remains intact, and will be overwritten when the granule is next used.) This general erasure of the granule allocation information makes killed disk files very difficult to reconstruct. For more information, consult the section below entitled "Retrieving Killed Files."

D I S K   R E P A I R

Introduction

If you encounter an error while using your disk you will know about it since you will either not be able to read from or write to the disk. When this happens you may have two options. If you are lucky enough to have a backup copy of your disk you may be able to backup your backup copy using the Backup command from BASIC. If you do not have a backup copy, or your backup copy is also crashed or unreliable, you then must consider repair. Disk repair requires that you use certain tools and that you follow a sequence of tests to determine and then fix the error.

The tools you will need are these: first and foremost, one or more formatted disks on which to write any files or sectors as may be required; a sharp pencil and a notebook to scrupulously record the nature and location of any error and to outline the steps for repair; and any or all of the following - a pot of coffee, your favorite brew, a good dose of patience and maybe something handy to smash.

Errors - Types, Causes and Repairability

Errors and their correction are at the heart of the VIP Disk-ZAP. Thus, a proper diagnosis of the error encountered is essential to proper disk repair. There are several ways to classify errors. The most essential classification is between those repairable and those not repairable. Another classification is between "hard" and "soft" errors. A third classification is according to the error messages that appear on the screen when a read or write is blocked because of an error.

Repairability is a fluid classification. You won't always know that something cannot be repaired until you

have tried everything and cannot make a repair. Sometimes you will know that the error is due to damage to the disk and that repair is not worth attempting. For example, you dropped your disk on your barbeque grill, or spread mayonaise on it. In such cases you need only consider whether or not to salvage what you can from the disk.

The "hard/soft" classification refers to errors due to disk damage or defect vs. errors due to operating system failure. Errors emanating from disk damage or defect are usually not repairable. Disk damage includes defective manufacture, scratches, creases, spilling things on the disk, punctures, or a heavy dose of static electricity. Errors emanating from operating system failure may or may not be repairable, depending on the type of error.

One cause for operating system error which you should always look for is Bad Connections. A bad connection between your disk drive and your computer is usually caused by dirty connections. It can lead to almost any kind of error message since it causes faulty reads and writes of all sorts. This is a problem which should always be looked for when errors are erratic, and different error messages occur when you retry the read or write. The fix is with the hardware. First turn off your whole system. Next, erase the connecting plugs with a large pencil eraser, or jiggle the connectors to get a better connection. Now turn on your system to see if you still get the errors. If so, try cleaning again. If it still continues, and there is nothing wrong with your disk, there may be something wrong with your drive, disk controller or computer.

Error Prompts

When you are Verifying the disk to determine the existence and nature of any errors, or you encounter an error when performing any other read from or write to the disk while using VIP Disk-ZAP, you may be presented

with one of several different error messages. These error messages, are listed below with most of their causes: NOTE: These messages are not supplied by the operating system, but derive from the Western Digital disk controller chip (the Tandy operating system translates almost all of them into the ubiquitous "I/O ERROR"). The operating system messages are generalized from these and will be of little use in diagnosing errors. That's why you purchased this program.

Error Messages During A Read

Drive Not Ready: This message has several causes, mostly relating to the disk drive somehow not being ready to operate. Usually this comes from the drive door not being closed or no disk being in the drive. It can also be caused, however, by a parity error, by a crashed directory, by a defective or damaged disk, or by an unformatted disk in the drive.

Record Not Found: This message indicates that the operating system could not find the specified drive, track and sector. This could be due to incorrect input of the numbers, faulty ID bytes, a crashed directory, or a damaged diwk.

CRC Error: This message indicates that an error has been found in the checksum which indicates that the sector located has been incorrectly saved by from one bit to all 256 bytes. The checksum fault may occur in the sector ID bytes or data bytes. This error is the easiest to correct.

Lost Data: This message indicates that the system has failed to read ever} byte of data. It is rarely encountered, but if it is, try again. If it continues to recur, your disk drive may need adjustment.

Data Request: This message indicates that the system is not receiving data for some reason.

Bad Data Address Mark: This error occurs when there is an error in the sector ID bytes making the sector unreadable.

Bad File: This error occurs on a read only and indicates that the directory is faulty. This error will be indicated by either an asterisk or an "R" next to a file when you call up the directory of your disk. See the section on the Directory command in Part I.

Error Messages During A Write

Error messages during a write are the same as the following during a read: Drive Not Ready, Record Not Found, CRC Error and Lost Data. In addition the following error messages occur only during a write:

Write Protect: This messages indicates that the system cannot write to a disk because it is write protected with a write protect tab over the slot. Check to see if you left the VIP Disk-ZAP master diskette in your drive or have put a write protected disk in the drive.

Data Request: This message indicates that the system is not able to send data for some reason.

Write Fault: This error indicates that the operating system or disk system made an error while writing. Try again. If this error message continues, your computer system may need readjustment or repair.

The Fixes

As you can see, most of the error messages do not define the cause of the error encountered or specify the method for repair. All the messages indicate is what has caused the system to be unable to read or write. The errors which are specific are: CRC Error and Write Protect. Don't forget that errors may not only be due to faulty disks, but may also be due to bad connections

between your drive, disk controller and computer, or to faulty equipment. This problem was discussed above.

The fixes for all the errors can be divided into "automatic" repairs, minor repairs and modifications, and disk rebuilding. The latter two can sometimes be quite time consuming. Fortunately, by far the most often enountered error is the CRC Error, and it is the easiest one to correct - nearly automatic.

Automatic Repairs

Most errors you will encounter will be repairable "automatically". What is meant by "automatic" is that the errors can be corrected by rewriting the bad sector back to itself. The error messages which can be fixed this way are: CRC Error, Drive Not Ready and Record Not Found. The CRC Error can usually be fixed this way; the others less so.

CRC errors can have two general causes. Either the operating system just made one of its rare mistakes, or the disk sector with the CRC error has become "floosy." Unfortunately, a sector which is physically bad may not always appear bad. Sometimes it may act normal and may be written to; other times it will have a CRC error. Thus, although this "automatic" repair will be permanent for CRC errors caused by the operating system, it will not permanently repair CRC errors caused by a floosy disk. The proper repair for such a problem is to temporarily repair the bad sector, read the file off the disk and save it to another part of the disk under a different name, and then allocate the bad granule so that it cannot be used again (see granule allocation below).

To explain how to make an automatic repair it will be assumed that you have verified the disk and found the sector(s) containing this error. Using the Disk Zap function, use the <T> or <J> command to access the bad

sector. When found, go into the Modify function and do the following: press <BREAK> to write the sector to the disk, answer the DRV, TRK and SEC prompt with the number of the bad sector and press <ENTER>, answer the "Write or return" prompt with a <W>, give the drive, track and sector numbers for the write (press <ENTER> for the same ones), and answer the prompt "Do you really wish to write?" with a <Y> and the sector will be written back to the disk just as it is. This should correct the sector and you can continue to other errors, if any. You should try to read the sector to see if the fix worked.

The Drive Not Ready and Record Not Found errors may sometimes also succomb to this treatment. If this fix does not work you will have to try something else.

Minor Repairs and Modifications

Minor repairs and modifications cover many situations, which could involve your desire to modify a file created on one program for use in another program, or your need to make a minor correction to the GAT or directory entry sectors. Perhaps you wish to change a prompt in a program which you feel is irksome. Just use the Locate and Modify sections to find the irksome prompt in the program, ZAP a new prompt into place and write the sector back to the disk.

- Reallocating A Granule To Repair An I/O Error

One particularly helpful minor correction is reallocation of a bad granule so that it cannot be used by the system. When you save a file to the disk from BASIC, if the operating system encounters a sector which is physically bad, it will give you an I/O error. Everytime you try to save to that disk the same error will occur.

When this happens, you have to find the bad sector, and allocate the granule in which that sector resides so that the system will skip it. The way this is done is to zap a C0 hex into the granule allocation table in the position equated with the granule with the bad sector. You should recall from the discussion above about the GAT sector that the number C0 tells the system that this granule is the last granule in the file (the "C") and that no sectors are being used (the "0"). What this does is tell the system that this granule has already been used, so the system will skip it; and since it is the last one in the file, the system will not look for another granule related to it.

Here's what you do with the disk with the I/O error. First use the Verify function to find the bad sector. Record the exact location of the bad sector (track and sector number and granule number). Next, if not already done, convert the number you obtained into hexidecimal. The next step is to use the Enter Track command from the Disk Zap menu to call up the GAT table track 11 hex, sector 2 hex. Go into the Modify mode. The cursor will be at relative byte zero, equal to granule zero. The BYTE indicator will cue your byte location (be sure you have selected the hex input option so that the BYTE counter is in hex). Now move the cursor in the HEX display until the BYTE indicator is the same number as the granule number containing the bad sector that you obtained above. Be sure that the ASC flag is not on the screen so that you will be inputting into the HEX display (if it is, press <CLEAR>). Now - zap C0 into the byte in the HEX display at the position of the cursor and write the zapped GAT sector back to the disk.

Once you have reallocated the bad granule, you will again be able to write to and read from the disk. You will not, however, be able to backup the disk using the BASIC backup command since the sector is still bad. You should not, therefore, use this disk for much longer; instead you should copy its files to another disk and

discard it. If you do desire to backup the disk, you may do so with the Read Sectors and Write Sectors commands from the Master Menu. Read the following section.

- Backup the Unbackupable

At times you will not be able to backup a disk with the BASIC backup command, such as when you have to reallocate a granule with a bad sector. You may use the Read Sectors and Write Sectors commands from the Master Menu to backup such a disk, or copy files off a bad disk. All you have to do is sequentially read the tracks from the bad disk, including the directory track, and write them to the same track on a good disk. Be sure to copy to the corresponding tracks on the good disk so that the directory will be correct!

When you're done copying, go into the GAT sector and de-allocate any granules allocated with a C0 because of a bad sector. To do this follow the instructions for allocation of a granule above except replace the C0 with an FF hex to tell the system that the granule is available.

- Fixing A Tokenized BASIC File With A Bashed Sector

BASIC programmers too are subject to the fates. Disk sectors can go bad, making a valuable program unloadable. Unless you have a backup copy, you may lose it. This section will help you recover most of your lost program.

As was mentioned above, BASIC programs are saved in a tokenized machine language form in that the BASIC commands have numeric equivalents (see BASIC Token chart in Appendix). BASIC files have a special format when saved. BASIC programs must be loaded to begin in a specific place in computer memory. They are stored in line units, each line unit pointing to the place in memory where the next line is located. Individual

program lines also have a set structure. Each file, and the first line in the file, begins with a 00 byte. (This and all future numbers in this discussion are in hex unless specified otherwise.) The following two bytes point to the location in memory where the next line is to begin. Following that are two bytes giving the line number of that line of the program. The contents of the program line then follow, terminated with a 00. The next line begins with the pointer bytes, then the next line number, and so on. The actual contents of the program line are tokenized basic commands and data strings.

To illustrate, a short three line program will be explained. The first program line "10 PRINT MEM" would look like this in memory:

00 26 09 00 0A 87 FF 93 00

It is assumed for this example that the starting memory location of the program is 2600 (again, all these numbers are in hex). The first 00 tells us that this is the beginning of the program in the file. This first line is nine bytes long. The second and third bytes, 26 09, are the pointer bytes. They tell the computer to go to memory location 2609 to get the next line of the program. The next two bytes, 00 0A, are the line number, a decimal 10. The next byte, 87, is the tokenized form of the PRINT command of BASIC; and the next two bytes, FF 93, are the tokenized form of the MEM command (for a list of the tokens see the Appendix). The final 00 tells the computer that this is the end of the line, and that it should consult the previous pointer bytes to find the next program line.

The next program line is: 20 A$ = INKEY $. This is represented in memory as 26 13 00 14 41 24 B3 FF 92 00. The first byte of this line, 26, is located at memory location 26 09. The 26 13 is the pointer, telling the computer to go to that location in memory for the start of the next line. The 00 14 is the line

number (decimal 20); and so on to 00, the end of line marker. The last line of this little program is:
30 A = PEEK(0), whose hex display is: 26 1F 00 1E 41 B3 FF 86 28 30 29 00 00 00

This last line begins at 26 13 and points to 26 1F. Again, there is a line number, and tokenized commands, and an end of line 00. If you count using all sixteen fingers, starting at byte 26 at memory location 26 13, you will see that the 26 1F points to a 00 and it is followed by a 00. This indicates that the file is ended. Now you know how to find your way through a BASIC file.

So what happens when a sector of your valuable program is bashed? Well, first, BASIC will not be able to read it so you will have to inspect it using the VIP Disk-ZAP. When you look you will probably be able to locate the beginning of the line and maybe the pointer, but the contents will be bashed - maybe the whole sector will be garbage. One sector is lost. The object is to save what you can of the program so that you don't lose it all.

The first step is to use the Read and Write Sectors command to save the whole program, every sector in every granule allocated to it, and the whole directory track, to a good disk. Once you have done this, begin the operation. Just to tell you now, our goal will be to load the program back into BASIC having the system ignore the bashed sector. This will require that you do two things. You first must change the pointer in the last good sector which originally pointed to the line(s) in the bashed sector, so that it now points to the first line in the first good sector after the bashed sector. Secondly, you must establish a bogus line number at the beginning of the bashed sector so the system will load it and so you can easily delete the garbage.

The first task is as simple as it sounds. Go to the first good sector prior to the bashed sector and

look for the last 00 byte before the end of the sector. This tells you a line ends there. Since BASIC lines must be less than 240 bytes, there has to be at least one 00 byte in every sector. Now look for the beginning of that line (which may be in the next sector back). When you find it (just after the next previous 00 byte) you will be looking at the pointer bytes. Be sure to note the contents and location of these. Also note the line number from the third and fourth bytes after the 00. Next, go to the first good sector after the bashed sector and do the same thing: locate the first 00 in the sector. This ends a bad line which emanates from the bashed sector. The two bytes after the 00 are again the pointer bytes; the third and fourth bytes are the line number. Note this line number.

Now that you have the line numbers and know where the 00's are you can repoint. To repoint you must add the number of bytes between the two good lines to the number in the pointer bytes of the last good line before the bashed sector. Thus, start from the last 00 before the bashed sector and count the bytes to the end of the sector; next add to this the 256 bytes from the bashed sector; finally add the number of bytes from the bashed sector to the first byte of the pointer after the first 00 in the next good sector. Now you have a decimal number. Convert it to hex with the chart in the appendix and add it, in hex, to the hex number in the pointer of the last good line before the bashed sector. For addition try: PRINT HEX$ (&H XX XX + &H XX XX), where the XX's are your hex numbers. The sum is the number to be zapped into the pointer in the last good line before the bashed sector. Write this sector back to itself to complete the zap. Now, when the program is loaded, the system will go from that good line to the memory location of the start of the first good line after the bashed sector.

Now only one last thing. Go to the bashed sector, and see if you can find a 00 followed by a pointer and line number. If so, you need not do any zapping. If

not, in the first two byte locations in the bashed sector zap in a valid line number and then a 00. To be valid, the line number must be greater than the last good line number before the bashed sector and less than the next good line number. Check the line numbers you noted down to be sure.

Once you have completed these repairs, load the program into BASIC. It should load nicely, except that the bashed sector should load as garbage. To get rid of it, type in the line number you found or zapped into the bashed sector and press <ENTER>. When you list the program the garbage will be gone. Now all you need do is reconstruct the program line and resave the program.

See how easy it was!

Rebuilding Files, Tracks, Etc.

All along this manual has been concerned with ASCII files. That emphasis is even stronger in this section. Reconstructing BASIC files not saved in the ASCII format and binary object code files are for the experienced only; we cannot hope to teach you how to reconstruct binary files.

Rebuilding files, you should know it now, is REAL WORK. It is a last resort when all else has failed and you absolutely must have the data stored on the disk. The degree to which you will succeed will depend primarily on your patience and planning. You will have to carefully record contents of sectors and plan how you will rebuild your files.

The primary reason you will ever need to rebuild a file is that somehow your directory track has been altered, either by disk damage, operating system failure, or accidental killing of a file. Directory failure should be checked whenever you receive the Drive Not Ready, File Not Found or Disk Bad messages or a CRC error in the directory, and all other sources for the error have been ruled out. The error may be in the GAT sector, the directory entry sectors or both.

To determine where the errors are, go into Disk Zap and access track 17, sector 3 (the first directory entry sector). (If a CRC Error occurs, try the CRC fix above.) See if it contains any garbage. If all looks OK there, continue through to the end of directory entry sectors with file entries. If the directory entry sectors are OK, and even if the directory sectors are filled with junk, go to sector two of track 17, the GAT sector, and see if that sector is intact.

Checking to see if the GAT sector is intact is not the easiest thing ever done. Of course, if the whole sector is filled with junk, it obviously is bad since only the first 68 of the 256 bytes are supposed to have

anything but FF's (255 decimal), the rest being orange graphics pixels (FF's).

If there seems to be the correct number of FF's, the job becomes more difficult since the other characters are ASCII representations of the numeric equivalents. Go into the modify mode to further check the numbers in the byte positions. If the HEX display contains any numbers other than FF, 00 to 43 or C0 to C9 (all in hex), some garbage has entered the GAT sector; this is also true if the same number, except FF and C0-C9, appears in more than one position.

Once you have determined that the directory has been damaged your FIRST task is to clone your directory to either another disk or to a free space on the same disk. Do this using the Read Sectors and Write Sectors commands from the Master Menu. This is done so that you can operate on your sick directory without worrying about destroying it forever.

Once the directory is cloned, you must then inspect the amount of damage done to the directory. You will need to take careful notes to make use of any useful information about granule allocation that the directory sectors have to offer. Your task will be to search the disk sector by sector, granule by granule to find the sectors of your files. Take careful notes to keep track of the locations of your files and which granules you have inspected. You will need to know the exact number and location of the sectors, the number of granules in the files, the number of sectors in the last granule of the files, and the number of bytes used by the file in the last sector of the last granule. NOTE: The search is made more difficult when you are using an older disk since it will probably be quite full, and you will probably have killed and replaced many files many times. This usually means that a several granule file will be put on granules located all over the disk.

Once you have the search data, your job is to either recreate the bashed directory, if it is only slightly damaged, or recreate the files on the disk on a new disk. To recreate the directory, you must zap the pertinent granule, sector and byte information into the appropriate GAT and directory positions.

To recreate the disk files on a new disk, you must first have a formatted disk. Then do the following:

1) From BASIC, or from some other text editor such as the VIP Writer, create short, one-line files on the formatted disk with the old file names to save the trouble of zapping in all those filenames.

2) Copy the first granule of the file to the granule allocated in the directory entry.

3) Zap the number of an open granule into the granule position in the GAT sector of the granule just copied to.

4) Copy the next granule of the file to the granule on the new disk the number of which you just zapped into the GAT sector.

5) Continue so until you reach the last granule of the file. For the last granule ZAP a "C" plus the number of sectors used in the granule into the appropriate place in the GAT sector.

6) Count the number of bytes used by the file in the last sector and zap this number into relative bytes 14 and 15 of the directory entry for the file.

7) If some of the sectors contain bad data, you can correct them by zapping the correct data in place; or, if you don't know the correct data, you can replace the data with spaces (hex 20) or carriage returns (hex 0D) for easy detection and repair.

It is definitely going to take experience to master the technique of directory repair. Be sure to refer to your disk manual and other reference works for all the help you can get.

Retrieving Killed Files

Retrieving killed files is a form of rebuilding the directory. As was noted above, when a file is killed, the operating system does not erase your whole file; instead, it does two things. First, the initial byte of the file entry (somewhere on sectors 3 to 11 of track 17) is changed to 00. This causes the first character of the file name to be changed to 00. Second, the system goes to the GAT sector and changes all bytes corresponding to the granules of the file to FF (those nice orange blocks) indicating that those granules are not allocated and may be used. The actual data in the file granules remains intact on the disk until the granules are used when you save a new file to the disk. Your task will be to change those FF's back to the relative numbers of the granules corresponding to the granules wherein your file resides.

The first step is to go to the file entry in the directory and change the 00 in the first byte back to what it was (i.e., reconstitute your file name). The next step is to go to relative byte 13 to see where the first granule of your file is, and to relative bytes 14 and 15 to see how many bytes of the last sector of the last granule of the file are used by the file. Note these down.

Now to the GAT sector. If your file occupies only one granule, the job is easy. Using the number from relative byte 13 above, access that granule (remember to convert the granule number to the correct track and sector number - discussed above) and see how many sectors are used by that file. With this information, return to the GAT sector and go to the byte corresponding to the granule which the byte occupies,

and ZAP in a "C" and the number of sectors occupied by the file. That was simple.

It is where your file occupies more than one granule that the problems arise. You may still use the granule reference from relative byte 13 of the directory entry to find the first granule of the file, but from there you must search the disk for the rest of the file. There is a way to find your file on the disk without the laborious search techinique, but this requires that there be only one outstanding killed file so that the GAT is not absolutely littered with FF's. The first aid is the limited ability to find out the granule length of your file. This may be determined by looking at the GAT sector to see how many FF's are in between other valid numbers (00-43 and C0-C9 hex). Count the orange blocks. Of course, your file could have occupied granules after the last granules shown as allocated. This makes it difficult to determine the number of granules allocated, since the bytes corresponding to the kill granules are changed back to FF, just like the rest of the sector.

The second aid involves a more in-depth analysis of the GAT sector. By a careful analysis of the numbers of the granules allocated, preferably by writing them down on paper, you should be able to find some granules in the sequence from granule 0 to the last granule allocated that are not allocated. These are the granules to check first for your file.

Once you have found the locations of the granules occupied by the killed file, carefully rebuild the GAT sector, beginning with the byte corresponding to the number obtained from relative byte 13 from the directory entry. Remember, that byte in the GAT sector must point to the byte corresponding to the next granule of the file, and so on until you come to the byte corresponding to the last granule of the file, which must be zapped with a "C" plus the number of sectors of the granule used by your file. Then you're done. You've "unkilled" your file!

Standard ASCII Character Set

The standard ASCII character set, below, is comprised of the 128 ASCII symbols from decimal 0 through 127. The first column gives the first 128 ASCII symbols in order; the second column gives the display of those symbols using the non-ASCII option; the third column gives the display of those symbols using the ASCII option; the fourth column gives the decimal numeric equivalent of the ASCII symbol; column five gives the hexadecimal equivalent; and the final column explains how to generate the ASCII symbol and its screen display equivalent through your Color Computer keyboard while modifying a sector using the ASCII input option. You can see that to generate the ASCII symbol, you may usually either place the hex number in the appropriate location in the hex display in the hex option or, in the ASCII option, input a character from the keyboard in the proper location in the ASCII display. Note that several ASCII symbols may not be generated from the keyboard in the ASCII option, but may only be input in hex in the Modify mode

ASCII	NON-ASC DISP	ASC DISPLAY	DEC	HEX	KEY(S)
NULL	INVERSE @	GRAPHICS	0	0	HEX ONLY
CTRL A	a	INVERSE !	1	1	HEX ONLY
CTRL B	b	INVERSE "	2	2	HEX ONLY
CTRL C	c	INVERSE #	3	3	HEX ONLY
CTRL D	d	INVERSE $	4	4	HEX ONLY
CTRL E	e	INVERSE %	5	5	HEX ONLY
CTRL F	f	INVERSE &	6	6	HEX ONLY
CTRL G	g	INVERSE '	7	7	HEX ONLY
CTRL H	h	INVERSE (	8	8	HEX ONLY
CTRL I	i	INVERSE)	9	9	HEX ONLY
CTRL J	j	INVERSE *	10	A	HEX ONLY
CTRL K	k	INVERSE +	11	B	HEX ONLY
CTRL L	l	INVERSE ,	12	C	HEX ONLY
CTRL M	m	INVERSE -	13	D	HEX ONLY
CTRL N	n	INVERSE .	14	E	HEX ONLY
CTRL O	o	INVERSE /	15	F	HEX ONLY
CTRL P	p	INVERSE 0	16	10	HEX ONLY
CTRL Q	q	INVERSE 1	17	11	HEX ONLY
CTRL R	r	INVERSE 2	18	12	HEX ONLY
CTRL S	s	INVERSE 3	19	13	HEX ONLY
CTRL T	t	INVERSE 4	20	14	HEX ONLY
CTRL U	u	INVERSE 5	21	15	HEX ONLY
CTRL V	v	INVERSE 6	22	16	HEX ONLY
CTRL W	w	INVERSE 7	23	17	HEX ONLY
CTRL X	x	INVERSE 8	24	18	HEX ONLY
CTRL Y	y	INVERSE 9	25	19	HEX ONLY
CTRL Z	z	INVERSE :	26	1A	HEX ONLY
ESCAPE	[	INVERSE ;	27	1B	HEX ONLY
FS	\	INVERSE <	28	1C	HEX ONLY

STANDARD ASCII CHARACTER SET (cont.)

ASCII	NON-ASC DISP	ASC DISPLAY	DEC	HEX	KEY(S)
GS	]	INVERSE =	29	1D	HEX ONLY
RS	↑	INVERSE >	30	1E	HEX ONLY
US	<	INVERSE ?	31	1F	HEX ONLY
SPACE	SPACE	SPACE	32	20	<SPACE>
!	!	!	33	21	<!>
"	"	"	34	22	<">
#	#	#	35	23	<#>
$	$	$	36	24	<$>
%	%	%	37	25	<%>
&	&	&	38	26	<&>
'	'	'	39	27	<'>
(	(	(	40	28	<(>
)	)	)	41	29	<)>
*	*	*	42	2A	<*>
+	+	+	43	2B	<+>
,	,	,	44	2C	<,>
-	-	-	45	2D	<->
.	.	.	46	2E	<.>
/	/	/	47	2F	</>
0	0	0	48	30	<0>
1	1	1	49	31	<1>
2	2	2	50	32	<2>
3	3	3	51	33	<3>
4	4	4	52	34	<4>
5	5	5	53	35	<5>
6	6	6	54	36	<6>
7	7	7	55	37	<7>
8	8	8	56	38	<8>
9	9	9	57	39	<9>
:	:	:	58	3A	<:>
;	;	;	59	3B	<;>
<	<	<	60	3C	"<"
=	=	=	61	D	<=>
>	>	>	62	3E	">"
?	?	?	63	3F	<?>
@	@	@	64	40	<@>
A	A	A	65	41	<A>
B	B	B	66	42	<B>
C	C	C	67	43	<C>
D	D	D	68	44	<D>
E	E	E	69	45	<E>
F	F	F	70	46	<F>
G	G	G	71	47	<G>
H	H	H	72	48	<H>
I	I	I	73	49	<I>
J	J	J	74	4A	<J>
K	K	K	75	4B	<K>
L	L	L	76	4C	<L>
M	M	M	77	4D	<M>
N	N	N	78	4E	<N>

Standard ASCII Character Set (cont.)

ASCII	NON-ASC DISP	ASC DISPLAY	DEC	HEX	KEY(S)
O	O	O	79	4F	<O>
P	P	P	80	50	<P>
Q	Q	Q	81	51	<Q>
R	R	R	82	52	<R>
S	S	S	83	53	<S>
T	T	T	84	54	<T>
U	U	U	85	55	<U>
V	V	V	86	56	<V>
W	W	W	87	57	<W>
X	X	X	88	58	<X>
Y	Y	Y	89	59	<Y>
Z	Z	Z	90	5A	<Z>
[	INVERSE [	[	91	5B	<SFT><↓>
\	INVERSE \	\	92	5C	<ST><CL>
]	INVERSE]	]	93	5D	<SFT><→>
	INVERSE ↑	↑	94	5E	<↑>
←	INVERSE ←	←	95	5F	<SFT><↑>
	GRAPHICS	INVERSE @	96	60	HEX ONLY
a	INVERSE !	a	97	61	<a>
b	INVERSE "	b	98	62	<b>
c	INVERSE #	c	99	63	<c>
d	INVERSE $	d	100	64	<d>
e	INVERSE %	e	101	65	<e>
f	INVERSE &	f	102	66	<f>
g	INVERSE '	g	103	67	<g>
h	INVERSE (	h	104	68	<h>
i	INVERSE)	i	105	69	<i>
j	INVERSE *	j	106	6A	<j>
k	INVERSE +	k	107	6B	<k>
l	INVERSE ,	l	108	6C	<l>
m	INVERSE -	m	109	6D	<m>
n	INVERSE .	n	110	6E	<n>
o	INVERSE /	o	111	6F	<o>
p	INVERSE 0	p	112	70	<p>
q	INVERSE 1	q	113	71	<q>
r	INVERSE 2	r	114	72	<r>
s	INVERSE 3	s	115	73	<s>
t	INVERSE 4	t	116	74	<t>
u	INVERSE 5	u	117	75	<u>
v	INVERSE 6	v	118	76	<v>
w	INVERSE 7	w	119	77	<w>
x	INVERSE 8	x	120	78	<x>
y	INVERSE 9	y	121	79	<y>
z	INVERSE :	z	122	7A	<z>
{	INVERSE ;	INVERSE [	123	7B	HEX ONLY
|	INVERSE <	INVERSE \	124	7C	HEX ONLY
}	INVERSE =	INVERSE]	125	7D	HEX ONLY
	INVERSE >	INVERSE ↑	126	7E	HEX ONLY
Rubout	INVERSE ?	INVERSE ←	127	7F	HEX ONLY

Decimal-Hexidecimal Conversion Chart

DECIMAL	HEX	DECIMAL	HEX	DECIMAL	HEX
0	0	50	32	100	64
1	1	51	33	101	65
2	2	52	34	102	66
3	3	53	35	103	67
4	4	54	36	104	68
5	5	55	37	105	69
6	6	56	38	106	6A
7	7	57	39	107	6B
8	8	58	3A	108	6C
9	9	59	3B	109	6D
10	A	60	3C	110	6E
11	B	61	3D	111	6F
12	C	62	3E	112	70
13	D	63	3F	113	71
14	E	64	40	114	72
15	F	65	41	115	73
16	10	66	42	116	74
17	11	67	43	117	75
18	12	68	44	118	76
19	13	69	45	119	77
20	14	70	46	120	78
21	15	71	47	121	79
22	16	72	48	122	7A
23	17	73	49	123	7B
24	18	74	4A	124	7C
25	19	75	4B	125	7D
26	1A	76	4C	126	7E
27	1B	77	4D	127	7F
28	1C	78	4E	128	80
29	1D	79	4F	129	81
30	1E	80	50	130	82
31	1F	81	51	131	83
32	20	82	52	132	84
33	21	83	53	133	85
34	22	84	54	134	86
35	23	85	55	135	87
36	24	86	56	136	88
37	25	87	57	137	89
38	26	88	58	138	8A
39	27	89	59	139	8B
40	28	90	5A	140	8C
41	29	91	5B	141	8D
42	2A	92	5C	142	8E
43	2B	93	5D	143	8F
44	2C	94	5E	144	90
45	2D	95	5F	145	91
46	2E	96	60	146	92
47	2F	97	61	147	93
48	30	98	62	148	94
49	31	99	63	149	95

Decimal-Hexidecimal Conversion Chart (Cont.)

DECIMAL	HEX	DECIMAL	HEX	DECIMAL	HEX
150	96	186	BA	222	DE
151	97	187	BB	223	DF
152	98	188	BC	224	E0
153	99	189	BD	225	E1
154	9A	190	BE	226	E2
155	9B	191	BF	227	E3
156	9C	192	C0	228	E4
157	9D	193	C1	229	E5
158	9E	194	C2	230	E6
159	9F	195	C3	231	E7
160	A0	196	C4	232	E8
161	A1	197	C5	233	E9
162	A2	198	C6	234	EA
163	A3	199	C7	235	EB
164	A4	200	C8	236	EC
165	A5	201	C9	237	ED
166	A6	202	CA	238	EE
167	A7	203	CB	239	EF
168	A8	204	CC	240	F0
169	A9	205	CD	241	F1
170	AA	206	CE	242	F2
171	AB	207	CF	243	F3
172	AC	208	D0	244	F4
173	AD	209	D1	245	F5
174	AE	210	D2	246	F6
175	AF	211	D3	247	F7
176	B0	212	D4	248	F8
177	B1	213	D5	249	F9
178	B2	214	D6	250	FA
179	B3	215	D7	251	FB
180	B4	216	D8	252	FC
181	B5	217	D9	253	FD
182	B6	218	DA	254	FE
183	B7	219	DB	255	FF
184	B8	220	DC		
185	B9	221	DD		

BASIC Keyword Token Codes

DEC	HEX	BASIC KEYWORD	DEC	HEX	BASIC KEYWORD	(2 BYTE #)* DEC	HEX	BASIC KEYWORD
128	80	FOR	167	A7	THEN	255+128	FF+80	SGN
129	81	GO	168	A8	NOT	255+129	FF+81	INT
130	82	REM	169	A9	STEP	255+130	FF+82	ABS
131	83	'	170	AA	OFF	255+131	FF+83	USR
132	84	ELSE	171	AB	PLUS	255+132	FF+84	RND
133	85	IF	172	AC	MINUS	255+133	FF+85	SIN
134	86	DATA	173	AD	*	255+134	FF+86	PEEK
135	87	PRINT	174	AE	/	255+135	FF+87	LEN
136	88	ON	175	AF	↑	255+136	FF+88	STR$
137	89	INPUT	176	B0	AND	255+137	FF+89	VAL
138	8A	END	177	B1	OR	255+138	FF+8A	ASC
139	8B	NEXT	178	B2	>	255+139	FF+8B	CHR$
140	8C	DIM	179	B3	=	255+140	FF+8C	EOF
141	8D	READ	180	B4	<	255+141	FF+8D	JYSTK
142	8E	RUN	181	B5	DEL	255+142	FF+8E	LEFT$
143	8F	RESTORE	182	B6	EDIT	255+143	FF+8F	RIGHT$
144	90	RETURN	183	B7	TRON	255+144	FF+90	MID$
145	91	STOP	184	B8	TROFF	255+145	FF+91	POINT
146	92	POKE	185	B9	DEF	255+146	FF+92	INKEY$
147	93	CONT	186	BA	LET	255+147	FF+93	MEM
148	94	LIST	187	BB	LINE	255+148	FF+94	ATN
149	95	CLEAR	188	BC	PCLS	255+149	FF+95	COS
150	96	NEW	189	BD	PSET	255+150	FF+96	TAN
151	97	CLOAD	190	BE	PRESET	255+151	FF+97	EXP
152	98	CSAVE	191	BF	SCREEN	255+152	FF+98	FIX
153	99	OPEN	192	C0	PCLEAR	255+153	FF+99	LOG
154	9A	CLOSE	193	C1	COLOR	255+154	FF+9A	POS
155	9B	LLIST	194	C2	CIRCLE	255+155	FF+9B	SQR
156	9C	SET	195	C3	PAINT	255+156	FF+9C	HEX$
157	9D	RESET	196	C4	GET	255+157	FF+9D	VARPTR
158	9E	CLS	197	C5	PUT	255+158	FF+9E	INSTR
159	9F	MOTOR	198	C6	DRAW	255+159	FF+9F	TIMER
160	A0	SOUND	199	C7	PCOPY	255+160	FF+A0	PPOINT
161	A1	AUDIO	200	C8	PMODE	255+161	FF+A1	STRING$
162	A2	EXEC	201	C9	PLAY			
163	A3	SKIPF	202	CA	DLOAD			
164	A4	TAB	203	CB	RENUM			
165	A5	GOTO	204	CC	FN			
166	A6	GOSUB	205	CD	USING			

* The token codes for these BASIC keywords are double byte tokens as opposed to the others which are single byte tokens. The first byte of the double byte tokens is FF hex (255 decimal) and is the least significant byte; the second byte is the most significant byte.

How To Use Other VIP Library Programs

Each of the programs in the VIP Library, with the exception of VIP Speller and VIP Disk-ZAP, were specifically designed to create files compatible with other programs in the Library. With the Library you can perform the essential home business tasks and combine the results for many purposes.

The VIP Writer is one of the central programs in the Library. It contains the most sophisticated editing and printing features, and it is to be used to create all reports combining files created on other applicable Library programs. Its companion is the VIP Speller. The Speller is an indispensible tool to correct typos and misspellings in VIP Library files.

VIP Calc is used to create financial or mathematical reports. It contains sophisticated print functions for independent printing of such reports. You may create files usable by the VIP Writer for reports to be combined with other text, and you may create templates with the VIP Writer for use in VIP Calc.

VIP Terminal is a communications program capable of transmitting and receiving any ASCII file, including VIP Library files. ASCII files can be transferred to the VIP Writer for further editing. The Terminal program also allows you to transfer files to work, clubs or friends. You can also print files received from others.

VIP Database, similar to VIP Calc, has its own sophisticated print functions for independent printing of database files. You can also create files for use with the VIP Writer to create combined text and database files.

VIP Disk-ZAP is a disk repair utility designed to repair any kind of file created using the Color Computer disk operating system. Of course, it therefore will also work on other Library files.

NOTES